Electronic Notes in Theoretical Computer Science 16 No. 2 (1998)
URL: http://www.elsevier.nl/locate/entcs/volume16.html 14 pages

Deadlock Behaviour in Split and ST Bisimulation Semantics

J.C.M. Baeten

Department of Mathematics and Computing Science
Eindhoven University of Technology
Eindhoven, The Netherlands

J.A. Bergstra

Programming Research Group
University of Amsterdam
Amsterdam, The Netherlands
and Department of Philosophy
Utrecht University
Utrecht, The Netherlands

Abstract

We investigate split and ST bisimulation semantics, in particular the deadlock behaviour of processes in these semantics. We define and axiomatise a variant of ACP, where atomic actions and durational actions coexist, and define split and ST bisimulation semantics on this theory. We exhibit a closed term that has a deadlock in split semantics, but not in ST bisimulation semantics, and vice versa: a closed term that has a deadlock in ST bisimulation semantics but not in split semantics. As an application, we investigate different versions of durational communication.

1 Introduction

From the literature, see e.g. [12], [8], [11], we find that ST bisimulation semantics is a preferred non-interleaving semantics, due to several nice properties it has (such as refinement being a congruence). The axiomatisation of ST bisimulation semantics in [8] makes this semantics amenable for investigation in an axiomatically presented process algebra as ACP. This axiomatisation starts out by considering split semantics, another non-interleaving semantics.

In this article, we investigate split and ST bisimulation semantics in the setting of the process algebra ACP. We obtain an algebra in which atomic actions (for which the interleaving paradigm holds) and durational actions

(that allow concurrent behaviour) coexist. We propose notation and give an axiomatisation based on [8].

We stick to the viewpoint that the basic building blocks of the ACP process algebra are atomic actions. Atomic actions are truly atomic, i.e. (the observation of) the execution of an action is instantaneous. This is the basis of the interleaving paradigm (in a parallel composition, atomic actions from different components are interleaved or merged). Thus, atomic actions have no duration and cannot be refined. Besides these, we may want to give names to certain (sub)processes, and call these also actions, *durational actions*. Here, durational actions are defined as the sequential composition of two atomic actions, its beginning and its ending. Doing just that gives rise to split semantics, if we also ensure that an ending matches the corresponding beginning, then we obtain ST bisimulation semantics.

In the two theories, we investigate deadlock behaviour. We exhibit a closed term that has a deadlock in split semantics, but not in ST bisimulation semantics, and vice versa: a closed term that has a deadlock in ST bisimulation semantics but not in split semantics. In each case, we use purely equational reasoning, based on our axiomatisations, to verify the results.

We conclude there is a real difference between the two semantics, of fundamental consequence in the specification and verification of processes.

As an application, we investigate data transfer. Usually, we look upon communication as instantaneous, and use atomic actions to model this. Looking closer, communication can be durational, e.g., first a link is established, and then data is transfered. We consider a couple of alternatives modeled as durational actions, and find also here differing deadlock behaviour.

2 ACP with Priorities and No Exit Iteration

We start out from the well-known theory ACP [7], [5]. The signature of ACP contains the following ingredients:

- A given number of *core atomic actions* $r, s, c, t, u, \ldots$. Core atomic actions were introduced in [2]. CA is the set of core atomic actions. All core atoms are atomic actions, $CA \subseteq A$, where A is the set of all atomic actions $a, b, d, \ldots$. Later, we will construct atomic actions that are not core atoms.
- A constant *inaction* denoted δ. This constant is the neutral element of alternative composition. This process disallows termination, so can be used to denote deadlock behaviour.
- A binary operator *alternative composition* or *choice*, denoted $+$. Choice is resolved by the execution of an action.
- A binary operator *sequential composition*, denoted $\cdot$.
- A binary operator *parallel composition* or *merge*, denoted $\|$. In a parallel composition, one of the components executes an action, or more than one component execute an action together, a communication action. The merge

is axiomatised using two auxiliary operators: *left merge*, denoted $\between$, and *communication merge*, denoted $|$. Communication on atomic actions is given, subject to the conditions that communication is commutative and associative, and if one of the arguments is δ, the communication is δ. The communication function on atomic actions is considered a parameter of the theory.

- A unary operator *encapsulation*, denoted ∂_H, that blocks the execution of atomic actions from the parameter set H. It is used to encapsulate communicating actions from the environment.

The axioms of ACP are well-known, and given for easy reference in Appendix A.

The definition of an operational semantics by means of SOS deduction rules is also standard. By means of these rules, we define binary relations $. \stackrel{a}{\rightarrow} .$ and unary relations $. \stackrel{a}{\rightarrow} \surd$ on closed terms (for $a \in A$). Intuitively, they have the following meaning:

- $x \stackrel{a}{\rightarrow} x'$ means that x evolves into x' by executing a
- $x \stackrel{a}{\rightarrow} \surd$ means that x successfully terminates upon execution of a

We present the rules in Appendix A. We define (strong) bisimulation equivalence in the standard way, based on these rules. Since the rules are in *path* format, bisimulation is a congruence for all operators, and the set of closed terms modulo bisimulation turns into an algebra for the signature of ACP.

From [4], [5] we quote the result that the axiomatisation of ACP is sound and complete for the algebra of closed terms modulo bisimulation.

In the sequel, we talk about the deadlock behaviour of processes. For our purposes, the following definition suffices.

Definition 2.1 We define when a term has a deadlock. We give an operational definition and an equational definition. The equational definition is given in terms of basic terms. Since we know that all closed terms can be written as a basic term, this suffices. We refer to [1] for more information, and a proof that both notions coincide.

- Let t be a closed term. We say t *has a deadlock* if starting from t we can execute a number of steps to a process t' that cannot execute a step at all.
- Inductively, we define a set of closed terms DL as follows:

(i) For each atomic action a, $a \cdot \delta \in DL$

(ii) If $t \in DL$, and a is an atom and s any closed term, then $a \cdot t \in DL$ and $t + s \in DL$.

Now let t be a closed term. We say t *has a deadlock* if either t is δ, or there is a term $t' \in DL$ that equals t.

We will require two extra ingredients in the sequel: *priorities*, introduced in the ACP setting in [3], and *no exit iteration*, introduced in [9] as a spe-

cial case of the binary iteration operator of [6]. The priority operator θ is parametrized by a partial ordering on atomic actions (denoted $<$), and in a choice context, summands that start with an atomic action of high priority will block summands starting with an atomic action of lower priority. The axiomatization uses an auxiliary operator $\lhd$ called *unless*.

The no exit iteration operator $\cdot^\omega$ will execute its argument an unbounded number of times. There is no exit of the loop. We present the respective axiomatizations in Table 1. The axiom NEI is the defining axiom of no exit iteration, and the conditional axiom RSP$^\omega$ is the Recursive Specification Principle for no exit iteration, stating that restricted recursive equations of the form $y = x \cdot y$ have a unique solution, viz. x^ω. Note that our theory does not contain an empty or skip process, which ensures that no process can terminate immediately, and so these equations are all guarded.

$a \lhd b = a$ if $a \not< b$	P1	$\theta(a) = a$	TH1
$a \lhd b = \delta$ if $a < b$	P2	$\theta(x \cdot y) = \theta(x) \cdot \theta(y)$	TH2
$x \lhd (y \cdot z) = x \lhd y$	P3	$\theta(x + y) = \theta(x) \lhd y + \theta(y) \lhd x$	TH3
$x \lhd (y + z) = (x \lhd y) \lhd z$	P4		
$(x \cdot y) \lhd z = (x \lhd z) \cdot y$	P5	$x^\omega = x \cdot (x^\omega)$	NEI
$(x + y) \lhd z = x \lhd z + y \lhd z$	P6	$y = x \cdot y \quad \Rightarrow \quad y = x^\omega$	RSP$^\omega$

Table 1
Axioms for priority and no exit iteration ($a, b \in A \cup \{\delta\}$).

Also here, it is straightforward to give operational rules, see Table 2. The deduction rules for priority use so-called *negative premises*. Nevertheless, the rules are well-defined, and again induce an algebra of closed terms modulo bisimulation congruence. This result uses the fact that the rules are in so-called *panth* format, see [4]. In [4], we can also find the proof that the axioms for ACP plus priorities form a sound and complete axiomatisation for the model of closed terms modulo bisimulation equivalence.

In the case of no exit iteration, [9] proves the axioms given are sound for the model of closed terms modulo bisimulation equivalence. Fokkink also shows completeness for the restricted theory of BPA, that has no parallel composition or encapsulation. We leave as an open question, whether completeness also holds for the full ACP theory.

3 Durational Actions

Next, in order to introduce split and ST bisimulation equivalence, we will assume we also have actions that take time to execute, non-atomic actions. We will call these *durational actions*. To ease notation, we will assume that

$$\frac{x \xrightarrow{a} x',\ \forall b > a\ x \overset{b}{\not\to}}{\theta(x) \xrightarrow{a} \theta(x')} \qquad \frac{x \xrightarrow{a} \surd,\ \forall b > a\ x \overset{b}{\not\to}}{\theta(x) \xrightarrow{a} \surd}$$

$$\frac{x \xrightarrow{a} x',\ \forall b > a\ y \overset{b}{\not\to}}{x \triangleleft y \xrightarrow{a} x'} \qquad \frac{x \xrightarrow{a} \surd,\ \forall b > a\ y \overset{b}{\not\to}}{x \triangleleft y \xrightarrow{a} \surd}$$

$$\frac{x \xrightarrow{a} x'}{x^\omega \xrightarrow{a} x' \cdot (x^\omega)} \qquad \frac{x \xrightarrow{a} \surd}{x^\omega \xrightarrow{a} x^\omega}$$

Table 2
Deduction rules for priority and no exit iteration ($a, b \in A$).

for each core atom t, we have a durational action $\widehat{t}$. Alternatively, we write $dur(t)$.

Having introduced this new signature element, the question is how to extend the axiomatisation of ACP to include durational actions. In particular, we cannot have $\widehat{t} \parallel \widehat{u} = \widehat{t} \cdot \widehat{u} + \widehat{u} \cdot \widehat{t}$, so the semantics of durational actions will not be purely interleaving.

Here, we use the notion that each durational action is composed of two atomic actions, namely the beginning of the action followed by the end of the action. We denote this as follows:

$$\widehat{t} = t^0 \cdot t^1 \qquad (DUR)$$

In the operational semantics, we add the following extra deduction rule:

$$\widehat{t} \xrightarrow{t^0} t^1$$

where $t^0, t^1 \in A$ are new atomic actions (not core atoms). In the literature, we sometimes see t^+ instead of t^0 and t^- instead of t^1.

Adding either of these equations to ACP with durational actions gives us split semantics, as is well-known from [12], [8], [11].

Let us define exactly what we mean by the term *split bisimulation semantics*. First, we give the standard definition for the non-interleaving semantics of old ACP-terms, originally due to [13], next we give a definition on terms that may involve durational actions.

Definition 3.1 (i) If x is a closed ACP-term over the set of core atomic actions CA, then $\widehat{x}$ is this term with each core atom t replaced by $\widehat{t}$. Now let x, y be two closed ACP-terms. We define $x \sim_{split} y$ iff $\widehat{x} \leftrightarroweq \widehat{y}$, $\widehat{x}$ is bisimulation equivalent to $\widehat{y}$ (using the extra rule displayed above).

(ii) Let x, y be closed terms over the theory of ACP with durational actions. We define $x \approx_{split} y$, x is *split bisimulation equivalent* to y, if $x \leftrightarroweq y$ (using the extra rule displayed above).

In the sequel, we will always use the second definition.

The problem with split semantics that is often cited, is that it can be unknown which end matches a certain beginning. To illustrate the problem, consider the following term:

$$\widehat{r} \parallel \widehat{r} \cdot t \parallel \widehat{s} \cdot u \parallel \widehat{s}.$$

Suppose $\widehat{r} \mid \widehat{s} = \widehat{c}$ is a defined communication, which we take to mean that $r^0 \mid s^0 = c^0$ and $r^1 \mid s^1 = c^1$. Further, we have the regular communication $t \mid u = v$. Now, in the displayed term, we can execute a c^0 resulting from a communication between the first and third component and a second c^0 resulting from a communication between the second and fourth component. We expect that the communication v can only take place upon completion of both communications. However, we can "criss-cross" the communications, and execute a c^1 resulting from the second and third components, and later another c^1 resulting from the first and fourth components. After the first c^1, already v becomes possible, contrary to intuition. In the next section, we will see that such "criss-cross" behaviour can alter the deadlock behaviour of a process.

If, on the other hand, we always want to be able to match an ending with the correct beginning, we have to employ additional machinery in order to do so. The first axiomatisation that achieves this can be found in [14]. Here, we present an axiomatisation that is basically the one of [8] adapted to our setting.

We add to the signature a unary operator hps, the *history pointer shift*, with auxiliary operators hps_n for each $n \geq 0$. Further, for each core atom t, there are new atomic actions $t^{hp(n)}$ for $n \geq 0$ and durational actions $dur(t)$, or, alternatively, if no confusion arises, atomic actions t^n and durational actions $\widehat{t}$. The intuition behind the history pointer is, that it will count how many steps ago the beginning of an action to be completed has occurred. In split semantics, the history pointer shift operator will be just the identity.

We replace axioms CM2 and CM3 of ACP by the axioms CM2HP and CM3HP shown in Table 3 below, and add the other axioms in this table.

The SOS rules for the history pointer are straightforward. See Table 4. In it, we also show the rules for merge and left merge of ACP, that have to be adapted.

Now we have the following definition of ST bisimulation semantics, originally defined in [10]. This follows the definition of split bisimulation semantics above.

Definition 3.2 (i) Let x, y be two closed ACP-terms (not containing durational actions or history pointer). We define $x \sim_{ST} y$ iff $\widehat{x} \leftrightarrows \widehat{y}$, $\widehat{x}$ is bisimulation equivalent to $\widehat{y}$ (using the rules of Table A.2 and Table 4).

(ii) Let x, y be closed terms over the theory of ACP with durational actions and history pointer. We define $x \approx_{ST} y$, x is *ST bisimulation equivalent* to y, if $x \leftrightarrows y$ (using the rules of Table A.2 and Table 4).

Again, in the sequel we will only use the second part of this definition.

$$a \between x = a \cdot hps(x) \qquad \text{CM2HP}$$
$$(a \cdot x) \between y = a \cdot (x \parallel hps(y)) \qquad \text{CM3HP}$$

$$\widehat{t} = t^0 \cdot t^1 \qquad \text{DUR}$$
$$r^n \mid s^n = c^n \text{ if } r \mid s = c \qquad \text{CDUR}$$

$$hps(x) = hps_0(x) \qquad \text{HPS1}$$
$$hps_n(\delta) = \delta \qquad \text{HPS2}$$
$$hps_n(t) = t \qquad \text{HPS3}$$
$$hps_n(t^m) = t^{m+1} \text{ if } n < m \qquad \text{HPS4}$$
$$hps_n(t^m) = t^m \text{ if } n \geq m \qquad \text{HPS5}$$
$$hps_n(a \cdot x) = hps_n(a) \cdot hps_{n+1}(x) \qquad \text{HPS6}$$
$$hps_n(x + y) = hps_n(x) + hps_n(y) \qquad \text{HPS7}$$

Table 3
Axioms of durational actions with history pointer ($a \in A \cup \{\delta\}, t, r, s, c \in CA$).

$$\widehat{t} \xrightarrow{t^0} t^1$$

$$\frac{x \xrightarrow{t} x'}{hps_n(x) \xrightarrow{t} hps_{n+1}(x')} \qquad \frac{x \xrightarrow{t} \surd}{hps_n(x) \xrightarrow{t} \surd}$$

$$\frac{x \xrightarrow{t^m} x',\ m \leq n}{hps_n(x) \xrightarrow{t^m} hps_{n+1}(x')} \qquad \frac{x \xrightarrow{t^m} \surd,\ m \leq n}{hps_n(x) \xrightarrow{t^m} \surd}$$

$$\frac{x \xrightarrow{t^m} x',\ m > n}{hps_n(x) \xrightarrow{t^{m+1}} hps_{n+1}(x')} \qquad \frac{x \xrightarrow{t^m} \surd,\ m > n}{hps_n(x) \xrightarrow{t^{m+1}} \surd}$$

$$\frac{x \xrightarrow{a} x'}{x \parallel y \xrightarrow{a} x' \parallel hps(y)} \qquad \frac{y \xrightarrow{a} y'}{x \parallel y \xrightarrow{a} hps(x) \parallel y'} \qquad \frac{x \xrightarrow{a} \surd}{x \parallel y \xrightarrow{a} hps(y)}$$

$$\frac{y \xrightarrow{a} \surd}{x \parallel y \xrightarrow{a} hps(x)} \qquad \frac{x \xrightarrow{a} x'}{x \between y \xrightarrow{a} x' \parallel hps(y)} \qquad \frac{x \xrightarrow{a} \surd}{x \between y \xrightarrow{a} hps(y)}$$

Table 4
Deduction rules for history pointer ($t \in CA$).

Now we have the following proposition. The proof follows the recipe as given in [4].

Proposition 3.3 (i) *Let x, y be closed terms over the theory of ACP with durational actions. Then $x \approx_{split} y$ iff $x = y$ is derivable from the axioms*

of ACP and DUR, CDUR, i.e. the axiomatisation is sound and complete for the operational model.

(ii) *Let x, y be closed terms over the theory of ACP with durational actions and history pointer. Then $x \approx_{ST} y$ iff $x = y$ is derivable from the axioms of ACP, minus CM2, CM3, plus the axioms of Table 3. Again, the axiomatisation is sound and complete for the operational model.*

4 Deadlock in Split but not in ST

We prove that there exists a simple process, that deadlocks in split semantics, but not in ST semantics.

Proposition 4.1 *There is a closed term in the signature of ACP with durational actions and priority operator that has a deadlock in split semantics but not in ST bisimulation semantics.*

Proof. We use the communication function that has $r_i \mid s_i = s_i \mid r_i = c_i$ as the only defined communications on core atoms ($i = 1, \ldots, 9$). The set H contains all r_i, s_i actions, but no c_i actions, and the priority relation gives all core atoms priority over all other atomic actions.

We consider the following term:

$$\theta \circ \partial_H([\widehat{r_1} \cdot (r_2 r_3 + r_4 r_5) \between r_6] \parallel [\widehat{r_1} \cdot (r_2 r_7 + r_4 r_8) \between r_9] \parallel [(\widehat{s_1} \cdot s_2) \between (s_6 s_3 + s_9 s_7)] \parallel [(\widehat{s_1} \cdot s_4) \between (s_6 s_5 + s_9 s_8)])$$

After a couple of steps using the axioms in Table 3, we see that this term evolves into:

$$\begin{aligned}
&c_1^0 c_6(c_1^2 c_2 c_3 \cdot \theta \circ \partial_H([\widehat{r_1} \cdot (r_2 r_7 + r_4 r_8) \between r_9] \parallel [(\widehat{s_1} \cdot s_4) \between (s_6 s_5 + s_9 s_8)]) + \\
&c_1^0 c_9 \cdot \theta \circ \partial_H([r_1^4 (r_2 r_3 + r_4 r_5)] \parallel [r_1^2 (r_2 r_7 + r_4 r_8)] \parallel [s_1^4 s_2 \parallel s_3] \parallel [s_1^2 s_4 \parallel s_8]) + \\
&c_1^0 c_6(c_1^2 c_4 c_5 \cdot \theta \circ \partial_H([\widehat{r_1} \cdot (r_2 r_7 + r_4 r_8) \between r_9] \parallel [(\widehat{s_1} \cdot s_2) \between (s_6 s_3 + s_9 s_7)]) + \\
&c_1^0 c_9 \cdot \theta \circ \partial_H([r_1^4 (r_2 r_3 + r_4 r_5)] \parallel [r_1^2 (r_2 r_7 + r_4 r_8)] \parallel [s_1^2 s_2 \parallel s_7] \parallel [s_1^4 s_4 \parallel s_5]) + \\
&c_1^0 c_9(c_1^2 c_2 c_7 \cdot \theta \circ \partial_H([\widehat{r_1} \cdot (r_2 r_3 + r_4 r_5) \between r_6] \parallel [(\widehat{s_1} \cdot s_4) \between (s_6 s_5 + s_9 s_8)]) + \\
&c_1^0 c_6 \cdot \theta \circ \partial_H([r_1^2 (r_2 r_3 + r_4 r_5)] \parallel [r_1^4 (r_2 r_7 + r_4 r_8)] \parallel [s_1^4 s_2 \parallel s_7] \parallel [s_1^2 s_4 \parallel s_5]) + \\
&c_1^0 c_9(c_1^2 c_4 c_8 \cdot \theta \circ \partial_H([\widehat{r_1} \cdot (r_2 r_3 + r_4 r_5) \between r_6] \parallel [(\widehat{s_1} \cdot s_2) \between (s_6 s_3 + s_9 s_7)]) + \\
&c_1^0 c_6 \cdot \theta \circ \partial_H([r_1^2 (r_2 r_3 + r_4 r_5)] \parallel [r_1^4 (r_2 r_7 + r_4 r_8)] \parallel [s_1^2 s_2 \parallel s_3] \parallel [s_1^4 s_4 \parallel s_8]).
\end{aligned}$$

At this point, behaviour in the two semantics will start to diverge. For instance, in the second line, the first argument of the merge can communicate with the third, and the second with the fourth, if we obey the history pointers. In split semantics, also communication between the second and third argument, and between the first and the fourth argument, is possible.

In ST semantics, we obtain the following:

$$\begin{aligned}
&c_1^0 c_6(c_1^2 c_2 c_3 c_1^0 c_9 c_1^2 c_4 c_8 + c_1^0 c_9(c_1^4 c_2 c_3 c_1^5 c_4 c_8 + c_1^2 c_4 c_8 c_1^7 c_2 c_3)) + \\
&+ c_1^0 c_6(c_1^2 c_4 c_5 c_1^0 c_9 c_1^2 c_2 c_7 + c_1^0 c_9(c_1^4 c_4 c_5 c_1^5 c_2 c_7 + c_1^2 c_2 c_7 c_1^7 c_4 c_5)) +
\end{aligned}$$

$$+c_1^0c_9(c_1^2c_2c_7c_1^0c_6c_1^2c_4c_5 + c_1^0c_6(c_1^4c_2c_7c_1^5c_4c_5 + c_1^2c_4c_5c_1^7c_2c_7))+$$
$$+c_1^0c_9(c_1^2c_4c_8c_1^0c_6c_1^2c_2c_3 + c_1^0c_6(c_1^4c_4c_8c_1^5c_2c_3 + c_1^2c_2c_3c_1^7c_4c_8)).$$

We see there is no deadlock in ST semantics. In split semantics, we obtain extra summands due to criss-cross behaviour, and expansion gives:

$$c_1^+c_6(c_1^-c_2c_3c_1^+c_9c_1^-c_4c_8 + c_1^+c_9(c_1^-c_2c_3c_1^-c_4c_8 + c_1^-c_4c_8c_1^-c_2c_3+$$
$$+c_1^-c_4c_1^-c_2\delta + c_1^-c_2c_1^-c_4\delta))+$$
$$+c_1^+c_6(c_1^-c_4c_5c_1^+c_9c_1^-c_2c_7 + c_1^+c_9(c_1^-c_4c_5c_1^-c_2c_7 + c_1^-c_2c_7c_1^-c_4c_5+$$
$$+c_1^-c_4c_1^-c_2\delta + c_1^-c_2c_1^-c_4\delta))+$$
$$+c_1^+c_9(c_1^-c_2c_7c_1^+c_6c_1^-c_4c_5 + c_1^+c_6(c_1^-c_2c_7c_1^-c_4c_5 + c_1^-c_4c_5c_1^-c_2c_7+$$
$$+c_1^-c_4c_1^-c_2\delta + c_1^-c_2c_1^-c_4\delta))+$$
$$+c_1^+c_9(c_1^-c_4c_8c_1^+c_6c_1^-c_2c_3 + c_1^+c_6(c_1^-c_4c_8c_1^-c_2c_3 + c_1^-c_2c_3c_1^-c_4c_8+$$
$$+c_1^-c_4c_1^-c_2\delta + c_1^-c_2c_1^-c_4\delta)).$$

Here, the deadlocks in the first and the last terms represent the sub-process

$$\theta \circ \partial_H(r_5 \parallel r_7 \parallel s_3 \parallel s_8)$$

and the deadlocks in the middle two terms are due to the remainder

$$\theta \circ \partial_H(r_3 \parallel r_8 \parallel s_7 \parallel s_5).$$

□

We leave as an open problem, whether or not such an example can be found in a smaller signature. In our investigations, we found that both left merge and priority operator seem essential.

5 Deadlock in ST but not in Split

We prove that there exists a simple process, that deadlocks in ST semantics, but not in split semantics.

Proposition 5.1 *There is a closed term in the signature of ACP with durational actions, priority operator and no exit iteration operator, that has a deadlock in ST semantics but not in split semantics.*

Proof. We use the same communication function as in the previous section. The set H contains all r_i, s_i actions but in addition the action c_3. The core atoms have priority over other atomic actions within the scope of the priority operator. The following term shows deadlock behaviour in ST semantics but not in split semantics.

$$\partial_H(\theta(\widehat{r_1} \between r_2 \cdot r_2 \cdot {r_3}^\omega) \parallel \widehat{s_1} \cdot u^\omega \parallel s_2 \cdot \widehat{r_1} \parallel s_2 \cdot \theta(\widehat{s_1} \between {s_3}^\omega)).$$

In ST semantics, we see that this term equals

$$c_1^0c_2c_2c_1^0 \cdot \partial_H(\theta(r_1^3 \parallel {r_3}^\omega) \parallel s_1^3 \cdot u^\omega \parallel r_1^1 \parallel \theta(s_1^1 \parallel {s_3}^\omega)).$$

In the first and fourth argument of the merge, the actions along port 3 take precedence over the actions along port 1. However, communication along

port 3 cannot occur as this is blocked by the encapsulation operator. For the second and third argument, the history pointers do not match. Thus, in ST semantics, no further action is possible, so deadlock ensues. In split semantics, communication between the second and third term is still possible, after which an unbounded amount of u's is possible, so the result becomes

$$c_1^0 c_2 c_2 c_1^0 c_1^1 \cdot u^\omega.$$

□

Again, it is an open question whether an example can be found in a smaller signature. In the current example, we note the essential use of left merge, priority and iteration.

6 Durational Actions for Message Passing

As an application of the theory above, we sketch communication between two devices, when communication is not instantaneous but has a duration. In order to describe this, we need the *early read* and *process prefix* mechanisms of [2]. We briefly recall this mechanism. For the sake of this discussion, we assume a fixed finite set D of messages to be communicated.

First of all, regular communication assumes core atoms $r_i(d), s_i(d), c_i(d)$, for each communication port $i \in P$, message $d \in D$, with the only defined communication actions $r_i(d) \mid s_i(d) = s_i(d) \mid r_i(d) = c_i(d)$ on core atoms.

Next, we add a new set of atomic actions $er_i^B(v)$; ($i \in P$, v a placeholder ranging over B, and $B \subseteq D$), the *early read* atoms. The working of these can only be appreciated fully together with the *process prefix* operator ;. We display the axioms that we need, taken from [2], in Table 5. In this table, we use the notation $x[d/v]$ for the term x where all occurrences of v are substituted by d.

$er_i^B(v); x = \sum_{d \in B} r_i(d) \cdot x[d/v]$	PRE1
$a; x = a \cdot x$ otherwise, if $a \neq er_i^B(v)$	PRE2
$(x + y); z = x; z + y; z$	PRE3
$(x \cdot y); z = x; (y; z)$	PRE4

Table 5
Axioms of early read and process prefix ($a \in A \cup \{\delta\}$).

When we define deduction rules, we have to realize that the first deduction rule of Table A.2 also holds for the new atoms $er_i^B(v)$. Further, we have the rules in Table 6.

Now, we can imagine that in order for communication to take place, first the connection needs to be established. Then, communication becomes durational, and we can define the following new durational actions (here, r_i, s_i, c_i

$$\frac{x \xrightarrow{er_i^B(v)} x',\ d \in B}{x;y \xrightarrow{r_i(d)} (x';y)[d/v]} \qquad \frac{x \xrightarrow{a} x'}{x;y \xrightarrow{a} x';y}$$

$$\frac{x \xrightarrow{er_i^B(v)} \surd,\ d \in B}{x;y \xrightarrow{r_i(d)} y[d/v]} \qquad \frac{x \xrightarrow{a} \surd}{x;y \xrightarrow{a} y}$$

Table 6
Deduction rules for early read and process prefix ($a \in A, a \neq er_i^B(v)$).

are core atoms).

$$der_i^B(v) = r_i \cdot (er_i^B(v))^{hp(1)}$$
$$ds_i(d) = s_i \cdot (s_i(d))^{hp(1)}$$
$$dc_i(d) = c_i \cdot (c_i(d))^{hp(1)}.$$

Now, we consider the following process X. Take $D = \{0, \ldots, 9\}$ to be the set of digits, and $B = \{0, 2, 4, 6, 8\}$ the set of even digits. We have just one communication port i, and let H contain the $r_i, s_i, r_i(d), s_i(d)$ actions. Finally, P, Q, R are given processes.

$$X = \partial_H(der_i^B(v); P \parallel ds_i(5) \cdot Q \parallel ds_i(6) \cdot R)$$

The receiving process P only wants to accept even messages, so when the link with Q is made, deadlock ensues. We find:

$$X = c_i \cdot \delta + dc_i(6) \cdot \partial_H(P[6/v] \parallel ds_i(5) \cdot Q \parallel R).$$

On the other hand, we can consider the actions $dur(s_i(d)) = s_i(d)^0 \cdot s_i(d)^1$, $dur(c_i(d)) = c_i(d)^0 \cdot c_i(d)^1$ and $dur(er_i^B(v)) = er_i^B(v)^0 \cdot er_i^B(v)^1$. This last action is a bit strange, as v will be bound two times. In the following calculation, we cannot start with a communication between the first and the second term, as the durational read cannot accept a message 5.

$$Y = \partial_H(dur(er_i^B(v)); P \parallel dur(s_i(5)) \cdot Q \parallel dur(s_i(6)) \cdot R) =$$
$$= dur(c_i(6)) \cdot \partial_H(P[6/v] \parallel dur(s_i(5)) \cdot Q \parallel R).$$

We see again a difference in deadlock behaviour. It will be clear that the behaviour of action $dur(er_i^B(v))$ is peculiar. On the other hand, we see $der_i^B(v)$ as a reasonable programming primitive.

7 Conclusion

We investigated two non-interleaving process semantics, split semantics and ST bisimulation semantics, in the framework of the process algebra ACP. We found that atomic actions and durational actions can coexist. We give an easy-to-understand axiomatisation that is complete in either semantics.

We know that in some cases, the two semantics coincide, e.g. when there is no autoconcurrency. However, in general the two semantics are very different: there are closed terms that deadlock in one semantics but not in the other.

This has impact for the specification and verification of processes. Thus, if we want some of the advantages of a non-interleaving semantics, such as durational actions or refinable actions, then specifying or verifying a system in split semantics will not do, we need to consider the more difficult ST-bisimulation semantics.

We considered as an application the description of message passing by means of durational actions. Here again, we find a difference in deadlock behaviour, depending on the moment of variable binding.

References

[1] J.C.M. Baeten and J.A. Bergstra. Global renaming operators in concrete process algebra. *Information and Computation*, 78:205–245, 1988.

[2] J.C.M. Baeten and J.A. Bergstra. On sequential composition, action prefixes and process prefix. *Formal Aspects of Computing*, 6(3):250–268, 1994.

[3] J.C.M. Baeten, J.A. Bergstra, and J.W. Klop. Syntax and defining equations for an interrupt mechanism in process algebra. *Fundamenta Informaticae*, IX(2):127–168, 1986.

[4] J.C.M. Baeten and C. Verhoef. Concrete process algebra. In S. Abramsky, D.M. Gabbay, and T.S.E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 4, pages 149–269. Oxford University Press, 1995.

[5] J.C.M. Baeten and W.P. Weijland. *Process Algebra*. Number 18 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1990.

[6] J.A. Bergstra, I. Bethke, and A. Ponse. Process algebra with iteration and nesting. *The Computer Journal*, 37(4):243–258, 1994.

[7] J.A. Bergstra and J.W. Klop. Process algebra for synchronous communication. *Information and Control*, 60(1/3):109–137, 1984.

[8] N. Busi, R.J. van Glabbeek, and R. Gorrieri. Axiomatising ST-bisimulation semantics. In E.-R. Olderog, editor, *Proceedings of the IFIP TC2 Working Conference on Programming Concepts, Methods and Calculi (PROCOMET'94)*, number 56 in IFIP Transactions A, pages 169–188. North-Holland, Amsterdam, 1994.

[9] W.J. Fokkink. Axiomatizations for the perpetual loop in process algebra. In P. Degano, R. Gorrieri, and A. Marchetti-Spaccamela, editors, *Proceedings ICALP'97*, number 1256 in Lecture Notes in Computer Science, pages 571–581. Springer Verlag, 1997.

[10] R.J. van Glabbeek and F.W. Vaandrager. Petri net models for algebraic theories of concurrency. In J.W. de Bakker, A.J. Nijman, and P.C. Treleaven, editors, *Proceedings PARLE, Volume II*, number 259 in Lecture Notes in Computer Science, pages 224–242. Springer Verlag, 1987.

[11] R.J. van Glabbeek and F.W. Vaandrager. The difference between splitting in n and $n+1$. *Information and Computation*, 136(2):109–142, 1997.

[12] R. Gorrieri and C. Laneve. Split and ST bisimulation semantics. *Information and Computation*, 118:272–288, 1995.

[13] M. Hennessy. Axiomatising finite concurrent processes. *SIAM Journal on Computing*, 17:997–1017, 1988.

[14] M. Hennessy. A proof system for weak ST-bisimulation over a finite process algebra. Technical Report 6/91, University of Sussex, 1991.

A ACP

We list the axioms of ACP in Table A.1. In terms, we often omit the sequential composition operator $\cdot$. Of all operators, sequential composition binds the strongest, and alternative composition the weakest. The other operators are not ranked.

$x+y=y+x$	A1	$x \parallel y = x \between y + y \between x + x \mid y$	CM1
$(x+y)+z=x+(y+z)$	A2	$a \between x = a \cdot x$	CM2
$x+x=x$	A3	$(a \cdot x) \between y = a \cdot (x \parallel y)$	CM3
$(x+y)\cdot z=x\cdot z+y\cdot z$	A4	$(x+y) \between z = x \between z + y \between z$	CM4
$(x\cdot y)\cdot z=x\cdot(y\cdot z)$	A5	$(a\cdot x)\mid b=(a\mid b)\cdot x$	CM5
$x+\delta=x$	A6	$a\mid(b\cdot x)=(a\mid b)\cdot x$	CM6
$\delta\cdot x=\delta$	A7	$(a\cdot x)\mid(b\cdot y)=(a\mid b)\cdot(x\parallel y)$	CM7
		$(x+y)\mid z=x\mid z+y\mid z$	CM8
$\partial_H(a)=a$ if $a\notin H$	D1	$x\mid(y+z)=x\mid y+x\mid z$	CM9
$\partial_H(a)=\delta$ if $a\in H$	D2	$a\mid b=b\mid a$	C1
$\partial_H(x+y)=\partial_H(x)+\partial_H(y)$	D3	$(a\mid b)\mid d=a\mid(b\mid d)$	C2
$\partial_H(x\cdot y)=\partial_H(x)\cdot\partial_H(y)$	D4	$a\mid\delta=\delta$	C3

Table A.1
Axioms of ACP $(a,b,d\in A\cup\{\delta\})$.

We present the SOS rules for ACP in Table A.2. The rules for communication in lines 6 and 7 only hold in case $a\mid b$ is defined to be an atomic action, so not equal to δ. The rules are in the so-called *path* format, see e.g. [4], from which we know that the semantics induced by the rules have some nice properties.

$$a \xrightarrow{a} \surd$$

$$\frac{x \xrightarrow{a} x'}{x + y \xrightarrow{a} x'} \qquad \frac{y \xrightarrow{a} y'}{x + y \xrightarrow{a} y'} \qquad \frac{x \xrightarrow{a} \surd}{x + y \xrightarrow{a} \surd} \qquad \frac{y \xrightarrow{a} \surd}{x + y \xrightarrow{a} \surd}$$

$$\frac{x \xrightarrow{a} x'}{x \cdot y \xrightarrow{a} x' \cdot y} \qquad \frac{x \xrightarrow{a} \surd}{x \cdot y \xrightarrow{a} y}$$

$$\frac{x \xrightarrow{a} x'}{x \parallel y \xrightarrow{a} x' \parallel y} \qquad \frac{y \xrightarrow{a} y'}{x \parallel y \xrightarrow{a} x \parallel y'} \qquad \frac{x \xrightarrow{a} \surd}{x \parallel y \xrightarrow{a} y} \qquad \frac{y \xrightarrow{a} \surd}{x \parallel y \xrightarrow{a} x}$$

$$\frac{x \xrightarrow{a} x'}{x \between y \xrightarrow{a} x' \parallel y} \qquad \frac{x \xrightarrow{a} \surd}{x \between y \xrightarrow{a} y}$$

$$\frac{x \xrightarrow{a} x',\ y \xrightarrow{b} y'}{x \parallel y \xrightarrow{a \mid b} x' \parallel y'} \qquad \frac{x \xrightarrow{a} \surd,\ y \xrightarrow{b} y'}{x \parallel y \xrightarrow{a \mid b} y'} \qquad \frac{x \xrightarrow{a} x',\ y \xrightarrow{b} \surd}{x \parallel y \xrightarrow{a \mid b} x'} \qquad \frac{x \xrightarrow{a} \surd,\ y \xrightarrow{b} \surd}{x \parallel y \xrightarrow{a \mid b} \surd}$$

$$\frac{x \xrightarrow{a} x',\ y \xrightarrow{b} y'}{x \mid y \xrightarrow{a \mid b} x' \parallel y'} \qquad \frac{x \xrightarrow{a} \surd,\ y \xrightarrow{b} y'}{x \mid y \xrightarrow{a \mid b} y'} \qquad \frac{x \xrightarrow{a} x',\ y \xrightarrow{b} \surd}{x \mid y \xrightarrow{a \mid b} x'} \qquad \frac{x \xrightarrow{a} \surd,\ y \xrightarrow{b} \surd}{x \mid y \xrightarrow{a \mid b} \surd}$$

$$\frac{x \xrightarrow{a} x',\ a \notin H}{\partial_H(x) \xrightarrow{a} \partial_H(x')} \qquad \frac{x \xrightarrow{a} \surd,\ a \notin H}{\partial_H(x) \xrightarrow{a} \surd}$$

Table A.2
Deduction rules for ACP $(a, b \in A, a \mid b \in A)$.